Fall 2015

Midterm Exam Report

Yeojin Kim *Advance Algorithm Programming*

1 Summary of the two methods

Hedcuter method and voronoi method consists of two steps : constructing weighted voronoi diagram and creating disks(i.e. stippling). In case of constructing weighted voronoi diagram, initially both methods sample points in image pixels. Next, methods computes weighted voronoi diagram with given points using different approaches and move the points to new sites repeatedly. For both cases, basically the centroid of a region is defined as

$$\mathbf{C_i} = \frac{\int_A \mathbf{x}\rho(\mathbf{x})dA}{\int_A \rho(\mathbf{x})dA}. \tag{1}$$

1.1 hedcuter method

The idea of fast stippling[3] is mat the pixel to the stipple level and copy stipples in it while looping through pixels. For implementation, using the face that the two dimensional discrete function $f(x, y)$ of gray scale corresponds to the stipple level, it gathered the stipples looping through the site and propagating through the pixels. The step is as below:

1. Sample initial points $X = \{\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \cdots, \mathbf{x_n}\}$
2. Loop while the displacements don't exceed maximum site displacement and the number of iterations doesn't exceed maximum number of iterations,
   (a) Map the image value at $\mathbf{x_i}$ to stipple level $t_{\mathbf{x_i}}$
   (b) Propagate the stipple level at $\mathbf{x_i}$ to its neighbor
   (c) Compute the centroids $\mathbf{C_i}$
   (d) Move each points $\mathbf{x_i}$ to its new centers of a cell $\mathbf{C_i}$
3. Create disks

First of all, hedcuter samples a uniformly distributed random point for whole area of given image and test if this random point has high probability based on gaussian distribution. If that is the case, the algorithm keeps this point as an initial point and this process is repeated until generating n points. In the loop, the algorithm keeps mapping image using the location of a point(which corresponds to the coordinates of a pixel) as unique identification. This mapping image contains the two dimensional discrete function $f(x, y)$ which also can corresponds to stipple level t at each points with the number of levels l. Hedcuter method first computes and stores stipple level t only at given points using Eq. 2.

$$t_{\mathbf{x}} = l - I(\mathbf{x}) \tag{2}$$

Next, it sorts the stipples with stipple level, x values, and y values in ascending order. Stipple level has a highest priority and y values has a lowest priority in sorting. Starting from a stipple which has the lowest stipple level, check the neighbors of the stipple. A stipple level t_{new} of a neighbor nb can be newly computed from the current stipple c based on Eq. 3.

$$t_{new} = t_c + t_{nb} \tag{3}$$

In case that a neighbor has a lower stipple level than stipple level it already has, hedcuter method keeps new stipple level and add this neighbor as stipple.After propagation of stipple level through all pixels, hedcuter method collect pixels which is affected by given points, so called coverage of a site. When computing the new center of a stipple, it compute weighted average of stipple levels in coverage. At last, hedcuter method generates disks with sites information from the loop. The radius of disk is determined by intensity and user-specified disk size. The size of disks is inversely proportional to intensity of the sites.

1.2 voronoi method

A flow of voronoi method basically follows Lloyd's method[2], which as follows :

Algorithm 1 Lloyd's method

1: **while** generating point $\mathbf{x_i}$ not converged to centroids **do**
2: Compute the Voronoi diagram of $\mathbf{x_i}$
3: Compute the centroids $\mathbf{C_i}$ using equation (1)
4: Move each generating point $\mathbf{x_i}$ to its centroids $\mathbf{C_i}$
5: **end while**

First, voronoi method samples points on the image randomly using Mersenne twister. Next, voronoi method computes the voronoi region of given sample points(Fig. 1(a)) using plane-sweep algorithm. After creating voronoi diagram of given points, the algorithm computes the centroid of a cell iteratively looping through all these points. For example, with a given cell in Fig. 1(b), voronoi method calculates the line which is the extension of voronoi edge and is called clipping line(Fig. 1(c)). When two end points of voronoi edge are $\mathbf{x_1} = (x_1, y_1)$ and $\mathbf{x_2} = (x_2, y_2)$, the equation of clipping line is

$$ax + by + c = 0 \ \ where\, a = -(y_1 - y_2),\ b = (x_1 - x_2),\ and\, c = x_1(y_1 - y_2) - y_1(x_1 - x_2). \tag{4}$$

In the same way, we can generate clipping lines for all voronoi edges in a cell.

To obtain the integration of density $\rho(\mathbf{x})$, voronoi method creates regular grid on a cell with the user-specified number of subpixels(Fig. 1(d)). Using the clipping lines(Eq. 4), we can test if a grid point is inside or outside of cell. If a grid point $\mathbf{x_g} = (x_g, y_g)$ satisfies $ax_g + by_g + c < 0$ for all the clipping lines, it locates inside of a cell. The result of test is shown in Fig. 1(e). The red points mean outside and the green points mean inside.
When grid points are inside, the algorithm obtains $\int_A \rho(\mathbf{x})dA$ and $\int_A \mathbf{x}\rho(\mathbf{x})dA$ with intensity $I(\mathbf{x_g})$

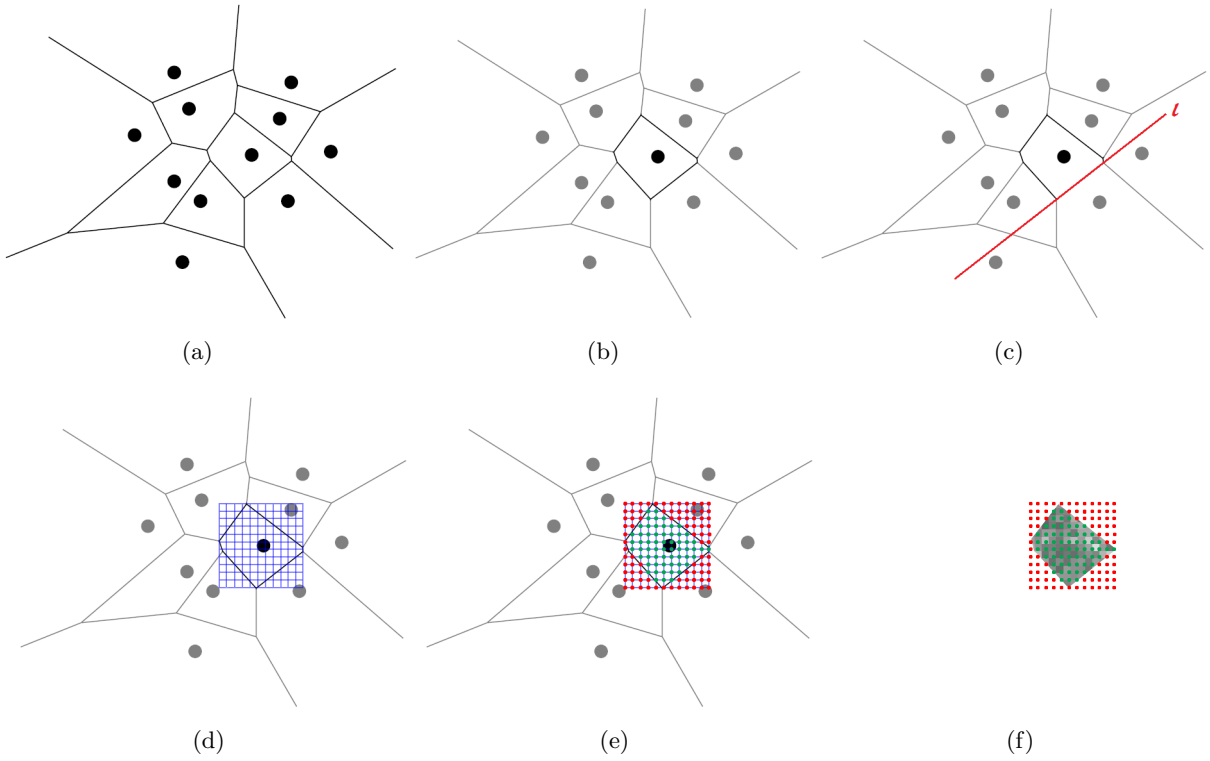


Figure 1: The progress of computing voronoi diagram and redistributing stipples.

which has corresponding location to grid points in grayscale image(Fig. 1(f), Eq. 5).

$$\begin{aligned} \int_A \rho(\mathbf{x})dA &= \int I(\mathbf{x_g}) \\ \int_A \mathbf{x}\rho(\mathbf{x})dA &= \int \mathbf{x}I(\mathbf{x_g}) = \int (x,y)I(\mathbf{x_g}). \end{aligned} \tag{5}$$

Substituting the results of Eq. 5 to Eq. 1, we can generate new centroid $\mathbf{C_i} = (x_c, y_c)$, which satisfies that $x_c = \frac{\int x \cdot I(\mathbf{x_g})}{\int I(\mathbf{x_g})}$ and $y_c = \frac{\int y \cdot I(\mathbf{x_g})}{\int I(\mathbf{x_g})}$. This new centroids will be fed as sites into creating voronoi diagram step. Voronoi method repeats creating voronoi diagram and computing the centroid repeatedly until the average of displacement between points $\mathbf{x_i}$ and centroids $\mathbf{C_i}$ is small enough. Finally, while creating the disks, voronoi method calculates the area of voronoi cell and computes the radius fits to this area.

2 Example Outputs

Here are the simple results of both methods with given examples(Fig. 2 and Fig. 3). As mentioned in the paper[3], the results of hedcuter method has some voids and overlapping stipples with same conditions. This is because of different densities at each stipple levels.
I also compare the output of two methods with various conditions in the following subsections.

2.1 same condition with multiple excution

Even if the initial condition is same, the result shows that hedcuter method generates different results at each time. With the same input image and the same number of initial samples, the points composed of squirrel head has different distribution in hedcuter method(the red squares in the first row images of Fig. 4). However, the results of voronoi method are exactly same(the second row images of Fig. 4). This is because two methods use different technique to generate samples. The hedcuter method uses random function with time seed(rand(time(NULL)), which produces different distributions when it calls. In case of voronoi method, it uses Mersenne twister(boost::mt19937) and it generates the same sample points with the same seed. Different initial points lead to different output in hedcuter method.

2.2 size of input image

Various size of input image brings interesting result(Fig. 5). From the left to the right column, the size of each image is 500x333, 750x499, 1500x997 and 3000x1993 with the same number of sample points. In case of voronoi method, the size is not an significant factor to stippling quality. However, hedcuter method generates more good quality stippling when the size of input image goes bigger. For example, the patterns on cat's head and whiskers(marked by blue square)becomes distinct and clustering of sample points(marked by red square) tends to disappear. This is caused by the nature of discrete voronoi diagram, as mentioned in [3]. If the resolution of image is lower, the relative error of the calculated centroid location is bigger. Voronoi method uses gaussian numerical integration to find the area with the density information and calculate the radius fits to this volume. This technique preserves the tone of stippling.

2.3 the number of disks

For both cases, if the number of disks increases, the result shows a tendency to draw details of original image. In Fig. 6, features such as the head of squirrel in hedcut method and the tail of squirrel in voronoi method(marked by blue square) shows increase of detail. One disadvantage of increasing initial samples is lower time performance(Fig. 7). There is a linear increase in time with the number of disks. Voronoi method shows the decrease of time with the number of disks because it uses multi-processing technique(Open MP) when it computes the tile and moves the sites. The area of each cell becomes smaller when the number of samples increases. Multiple smaller tasks advantage of parallel computation.

2.4 brightness of image

The result of different brightness is shown in Fig. 8. From the first column to the last column, the brightness of image is -40%, -20%, +20%, and 40% of original image in order. Generally if the input image goes brighter, its stippling result seems to draw the input image more clear. For example, the left side of cat's face(marked by blue square) distinguishes clearly from the background as the input image goes brighter. Whiskers of the cat are also good example. However, it is worth noticing that brightness can remove some important features such as the lower jaw of the cat(marked by

red square, Fig. 8(h), 8(k), and 8(l)). This is because features are blurred with brightness in the input image, not because of algorithm itself.

2.5 contrast of image

Following Fig. 9 shows the result of contrast. From the left to the right, the level of contrast of each column is -40%, -20%, +20% and +40% of the original image. Features in various contrast of image also affected in different way like brightness. Some features are blurred and some features are distinct. For instance, cat's eye in hedcuter method and the left side of face in voronoi method(marked by blue square) shows strong contrast in stippling results also. However, the left side of face in hedcuter method and the patterns on forehead shows dimmer results(marked by red square) while contrast grows stronger. This causes unexpected result of stippling.

3 Improvement of hedcuter method

3.1 distribution

Based on observation in Sec. 2, I concluded that the size of image can improve distributions. The number of disks also can improve distribution, but in hedcuter method case, it has linear increase in time without any acceleration technique. Brightness and contrast are uncontrollable, because some features are removed. Therefore I used subpixels to make virtual resolution image suggested in assignment paper.
The first step is to create high-resolution image of input when construction of voronoi diagram starts and map the sites to virtual image. Next the algorithm computes intensities based on virtual image with wider voronoi region and computes the new centroids. These cenetroids are mapped to original resolution. Here are results of simple gradient image with subpixels in Fig.10. It connect to the option **-subpixels** [**num**].
In first and second examples(rows) in Fig. 10, it hard to recognize how voids and clustering is reduced. In order to observed in detail, I made the third examples. It shows that when the number of subpixels increase, the top boundary reduces the clustering(marked by blue square). But the result shows that clustering still exists in Fig. 10(l) and it shows no improvement on distribution over the certain number of subpixels. Also if I increase the size of virtual resolution image, it need more time to compute the results.

3.2 time performance

While I observed the iteration of hedcuter method, I found that the sites moves walk around with the small distance, changing almost nothing of weighted voronoi diagram. Therefore, I add the average distance condition(**-avg**) for terminating the loop. This condition terminates when the average displacement of sites are less than user-specified value, not by max displacement. However, it works only the few iterations and brings nothing practical benefits and the user should defined the average distance.
Next attempt is to use GPU programming[1]. Before going into details, I want to report that this functionality is connected to option **-gpu** and this brings nothing meaningful results because I failed

to implement it. The idea is to draw cone mesh behind virtual plane and if we see the virtual plane from the top of it, the uncovered face of cones is voronoi region. The depth buffer automatically handle which points of cone is to be seen or not. For weighted voronoi diagram, all we needed is move the cone backward by the distance computed from intensity. If we draw cones with different color, we can read the color from pixel and obtain voronoi region. Next, we can compute the new centroids with this voronoi region and move the sites.
With OpenGL pipeline, simple reading pixels(using glReadPixel) of whole image for every loop makes the program stopped. glReadPixel function iteself is slow, therefore I need to use frame buffer with color and depth buffer to compute voronoi diagram following the paper[1] or using pixel buffers. After implementing buffers, I found the channels of image between frame buffer and what I had drawn in it is different for some reasons(I still don't know the reason why). So the result with this option is to show voronoi diagram of initial sites with native color.

3.3 extra functionality

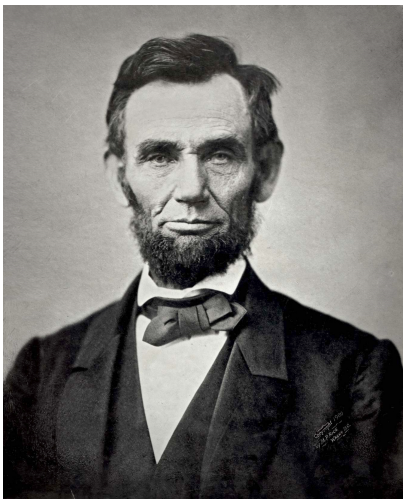
I add the functionality to generate colorful disks. Here are some results in Fig. 11. and Fig. 12. Colorful disks with sufficient number of samples generate good result such as Fig. 11(f) and Fig. 11(h). If we feed the algorithm with sufficient number of samples and make it generates the radius covered voronoi region enough, we can improve these kinds of examples(Fig. 12(b), Fig. 11(d), and Fig. 11(j)). At last, the effectiveness of color disks can be suffer from the input image in Fig. 12. If the quality of stippling affected by other factors, it might not express colors fully.

References

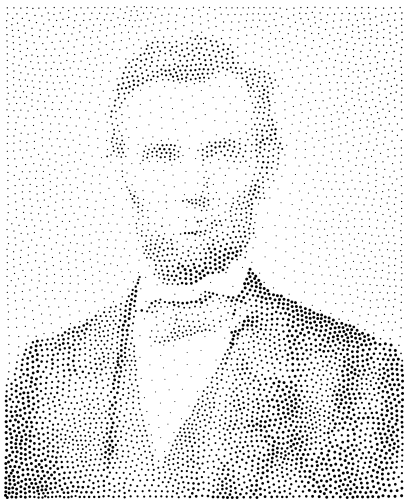
[1] Kenneth E. Hoff, III, John Keyser, Ming Lin, Dinesh Manocha, and Tim Culver. Fast computation of generalized voronoi diagrams using graphics hardware. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 277–286, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co. 3.2

[2] Atsuyuki Okabe, Barry Boots, and Kokichi Sugihara. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, Inc., New York, NY, USA, 1992. 1.2

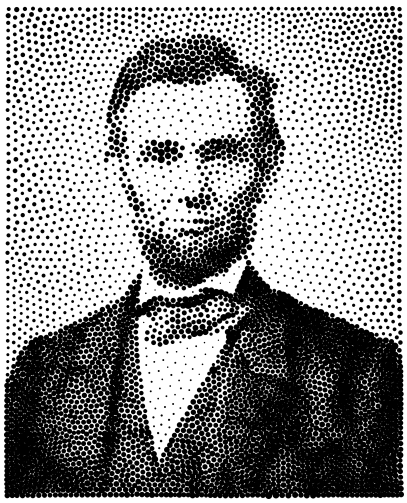
[3] Andrian Secord. Weighted voronoi stippling. In *2nd International Symposium on Non-Photorealistic Animation and Rendering (NPAR'02)*, pages 37–43, Annecy, France, June 3-5 2002. 1.1, 2, 2.2



(a)



(b)



(c)

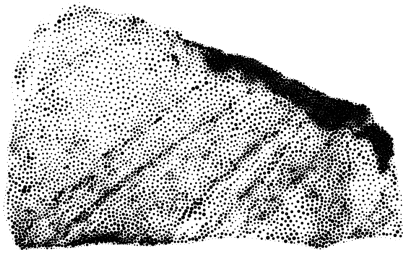
(d)

(e)

(f)

(g)

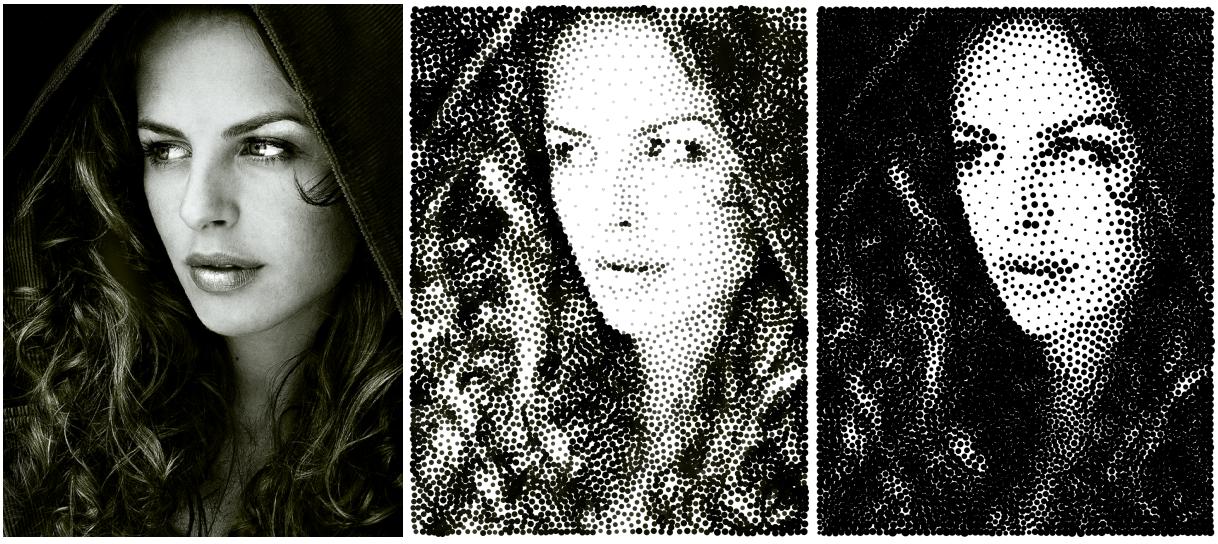
(h)



(i)

(j) (k) (l)

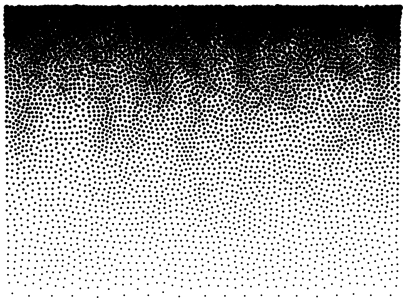
(m) (n) (o)



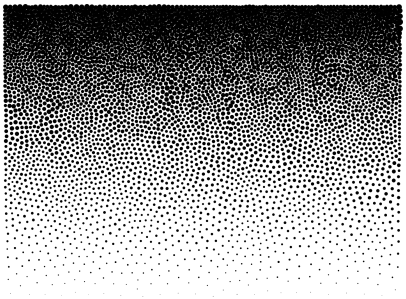
(p) (q) (r)

Figure 2: The simple output results of both methods. From the first column to the last column, it shows original images, the results of hedcuter method, and the results of voronoi method. Note that Fig. 2(j) and Fig. 2(m) has different sizes.

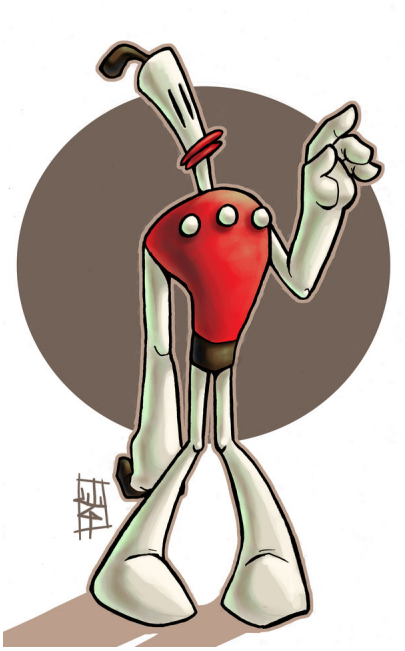
(a)



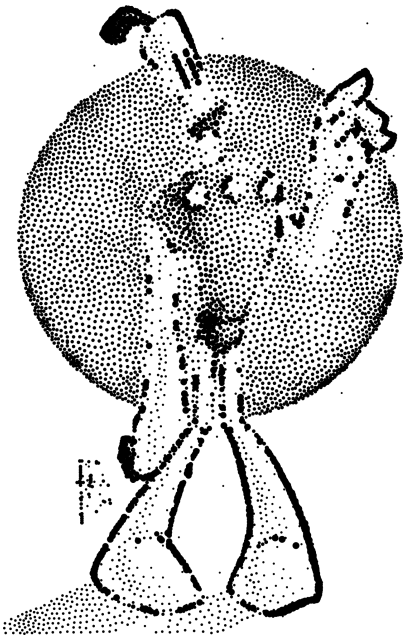
(b)



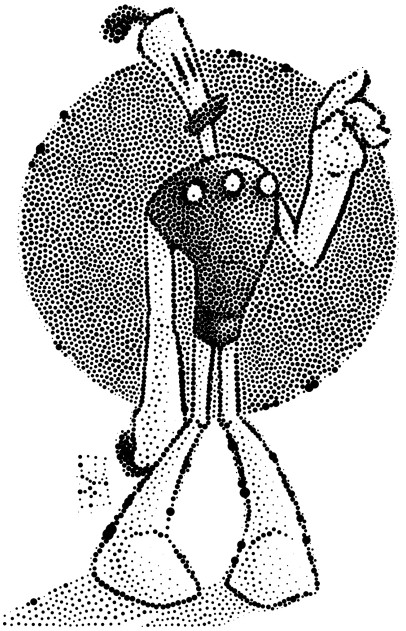
(c)



(d)

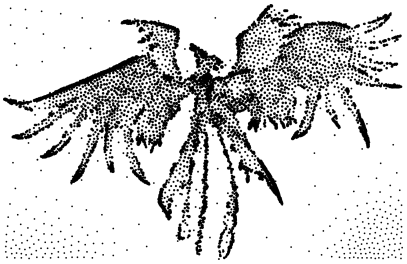


(e)

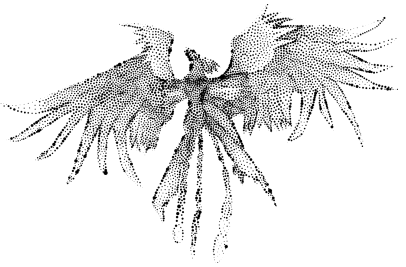


(f)

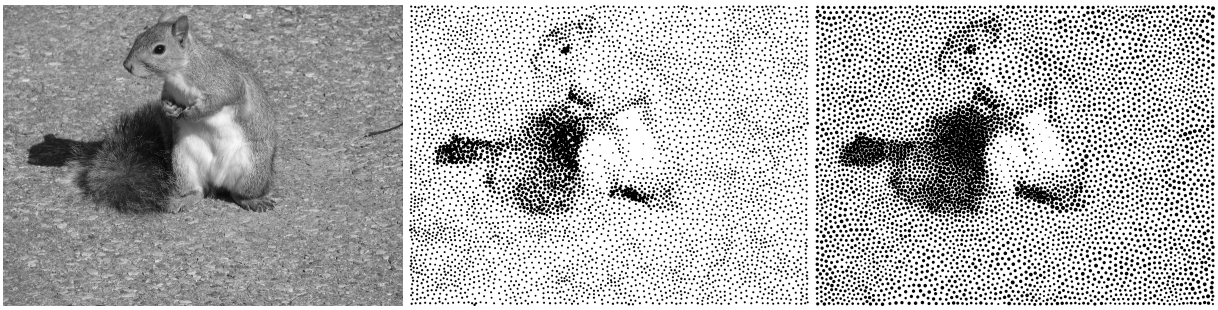
(g)



(h)

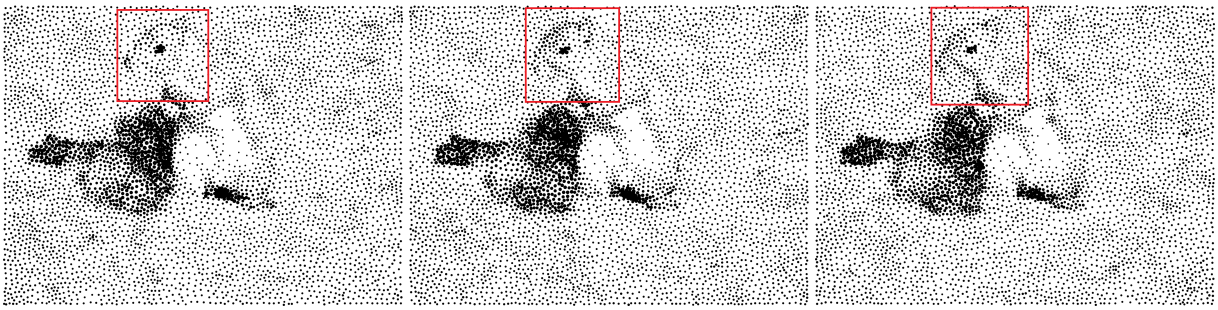


(i)

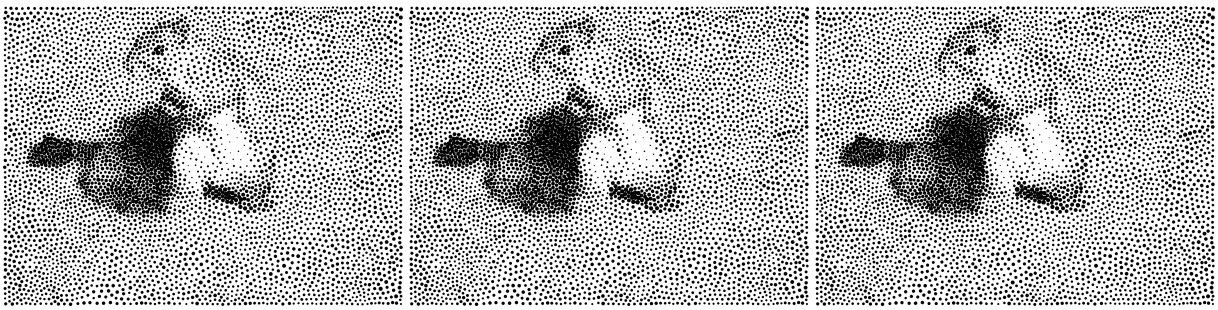


(j) (k) (l)

Figure 3: The simple output results of both methods. From the first column to the last column, it shows original images, the results of hedcuter method, and the results of voronoi method.

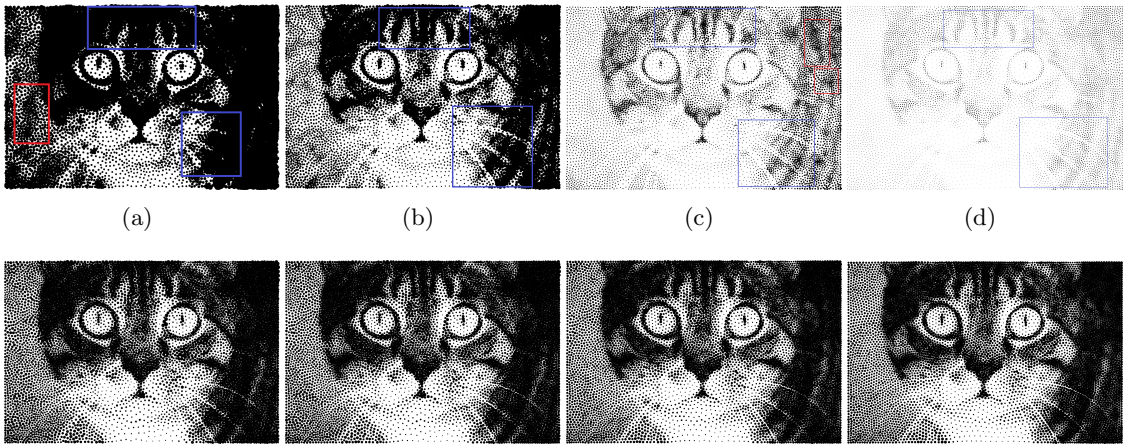


(a) (b) (c)



(d) (e) (f)

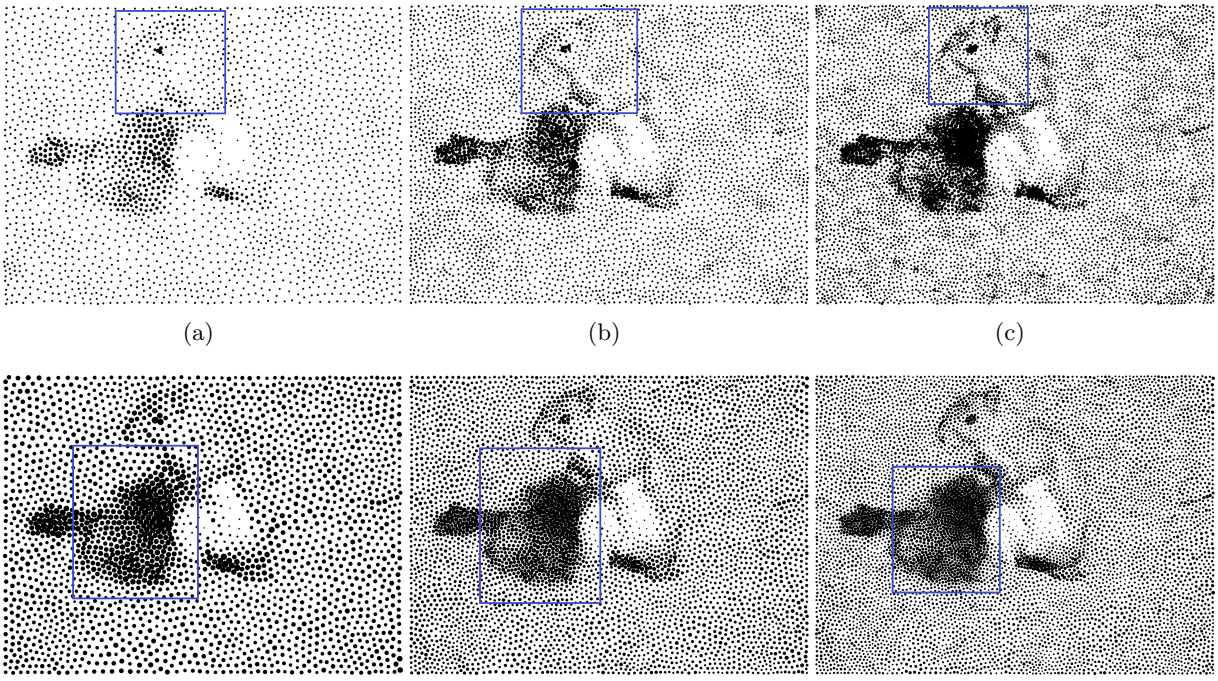
Figure 4: The output result of same initial condition for both methods. The first row is the result of hedcuter method, and the second row is the result of voronoi method.



(a) (b) (c) (d)

(e) (f) (g) (h)

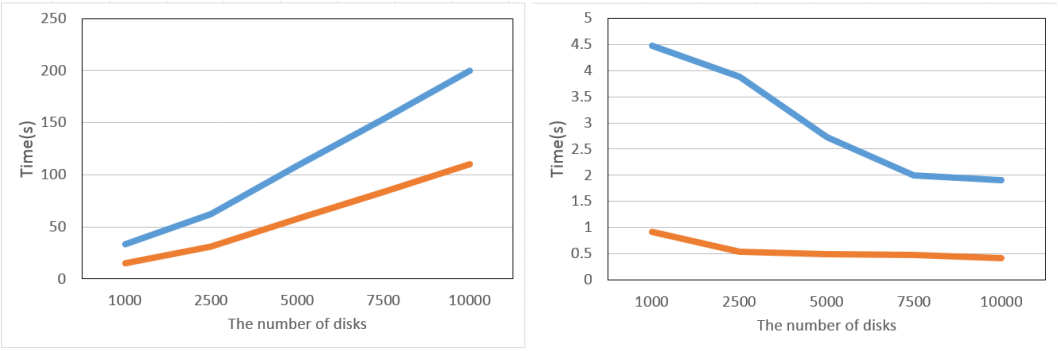
Figure 5: The output result of both methods with various size of input image. The first row is the result of hedcuter method, and the second row is the result of voronoi method.



(a) (b) (c)

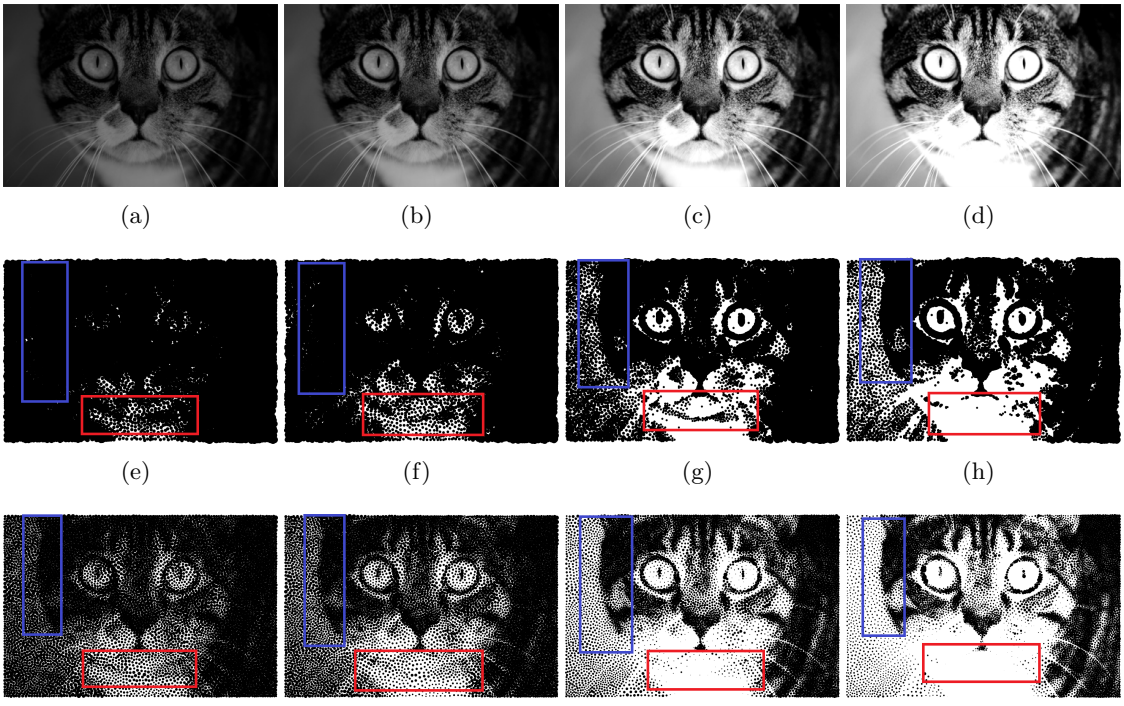
(d) (e) (f)

Figure 6: The output results of different number of disks. From top row is the results of hedcuter method, and bottom row is the results of voronoi method.



(a) (b)

Figure 7: The time performance of different number of disks. The left graph belongs to hedcuter method and the right graph belongs to voronoi method. The orange line is the result with Fig. 2(g) and the blue line is the result with Fig. 2(d).



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

Figure 8: The output results of different brightness. From top to bottom row, it shows input images, the results of hedcuter method, and the results of voronoi method.

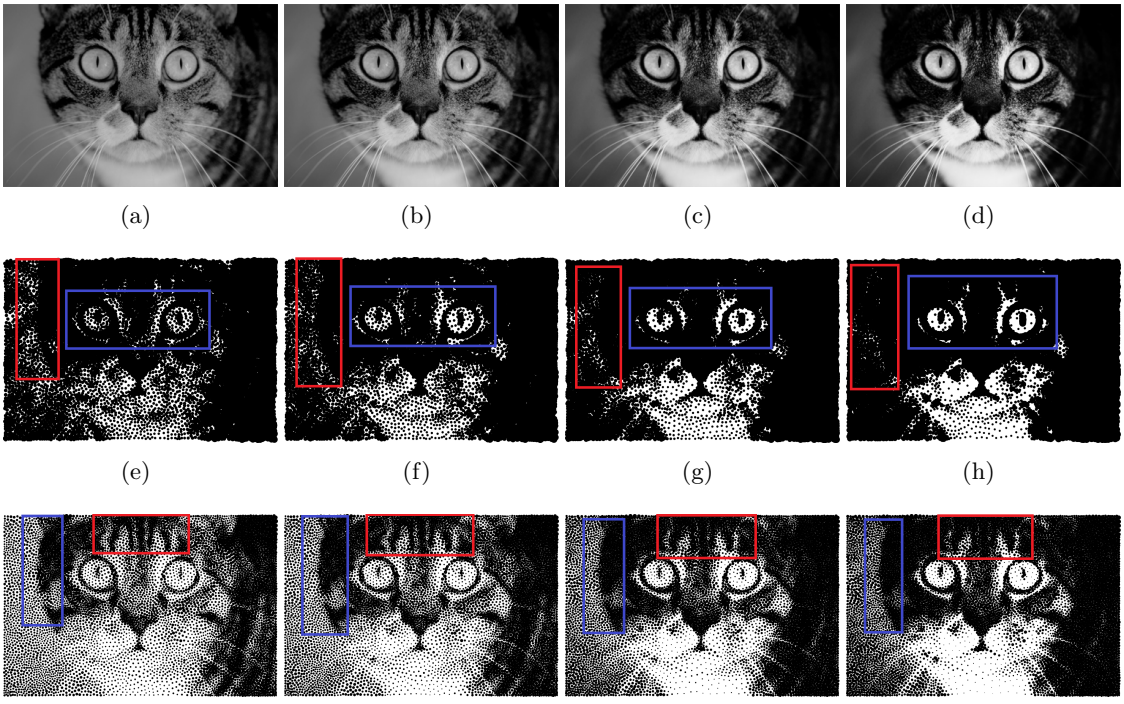


Figure 9: The output results of different contrast. From top to bottom row, it shows input images, the results of hedcuter method, and the results of voronoi method.

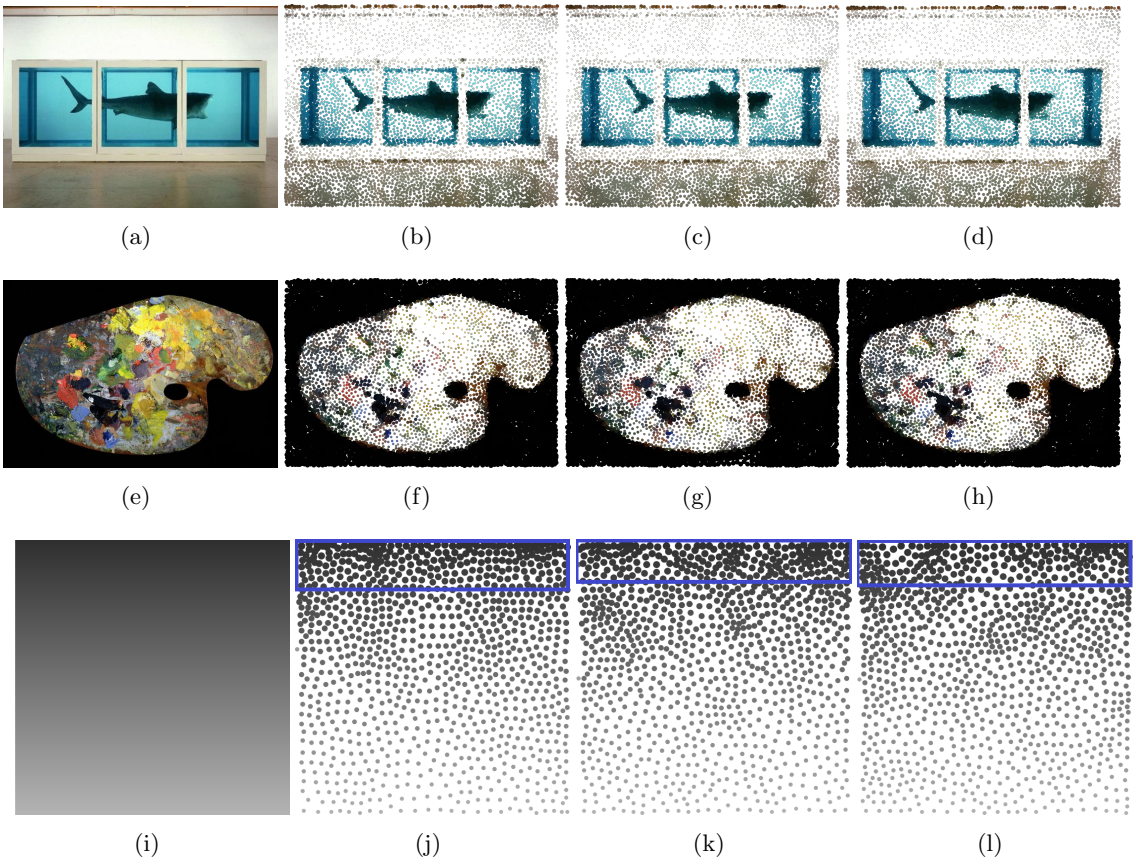
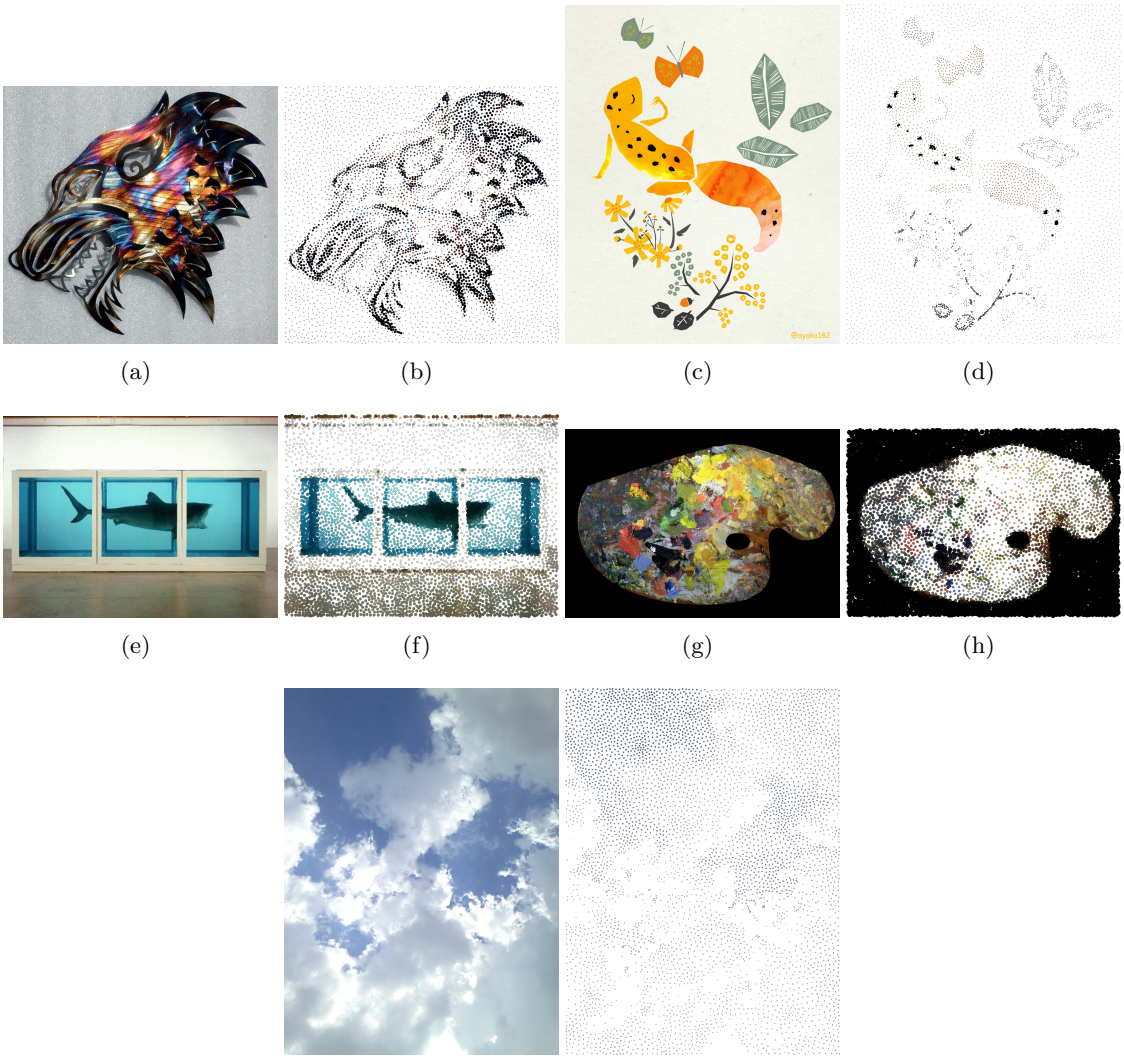


Figure 10: The output results of subpixels. The first column is the input image. From the second column to the last column, it shows the results with increasing subpixels.

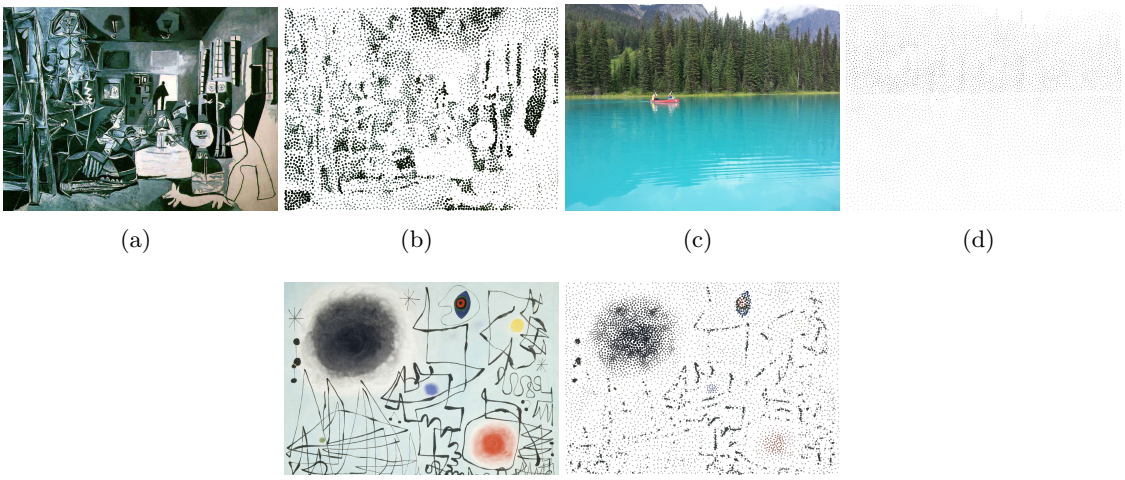


(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j)

Figure 11: The input image and good results with colorful disks.



(a) (b) (c) (d)

(e) (f)

Figure 12: The input image and bad results with colorful disks.